Energy-Aware Adaptive Framework for CAV

Katare, Dewant; Janssen, Marijn; Ding, Aaron Yi

DOI
10.1109/CCWC62904.2025.10903937

Publication date
2025

Document Version
Final published version

Published in
2025 IEEE 15th Annual Computing and Communication Workshop and Conference, CCWC 2025

Energy-Aware Adaptive Framework for CAV

Dewant Katare
Delft University of Technology
Delft, the Netherlands
d.katare@tudelft.nl

Marijn Janssen
Delft University of Technology
Delft, the Netherlands
m.f.w.h.a.janssen@tudelft.nl

Aaron Yi Ding
Delft University of Technology
Delft, the Netherlands
aaron.ding@tudelft.nl

***Abstract*—Driving assist applications and connected autonomous vehicle systems are supported using AI models and algorithms, which process and analyze heavy data volumes. High-performance computing units and large memory systems support these models, algorithms, and applications, which results in additional onboard energy consumption. The current trend is also towards full electrification of vehicles and increasing connectivity in the vehicular ecosystem to support collaborative and distributed applications using vehicle-edge-cloud computing. However, with the increased focus on model performance and improving the accuracy of these models and applications, the issue of high-performance computing requirements and resulting energy consumption are overlooked. The problem becomes more challenging and complex for resource-constrained edge devices, which are battery-dependent and have limited memory and computing power. This paper proposes components for an adaptive framework to reduce energy consumption by balancing model accuracy. The contributions include proposing and integrating model partition mechanisms, adaptive deployment across edge devices and approximation strategies for the models. By integrating these components, this framework supports energy-aware development across various platforms. The approach offers a sustainable method for computing and communication-oriented applications within the vehicular ecosystem.**

***Index Terms*—Energy-aware Computing, Model Approximation, ML Systems, Edge AI, Vehicle-Edge Computing**

I. Introduction

Autonomous vehicles have advanced with the integration of sensing units, intelligent algorithms and connectivity, advancing the traditional vehicle from a mode of transportation to a cyber-physical interconnected system [1]. These advancements use basic and complex computational models, enhancing vehicle navigation, efficiency, and user experience. In particular, Connected autonomous vehicles (CAV) use various AI models, including regression, classifiers, and complex attention mechanisms, to perform tasks that range from environmental sensing to decision-making and driver assistance functionalities using vehicle-edge-cloud computing ecosystem [2]–[5]. An overview of CAV is shown in Figure 1. These intelligent vehicles depend on real-time data processing and continuous communication, supported by high-performance computing units and large memory systems.

AI models in these systems manage tasks ranging from perception and actuation to navigation, necessitating ongoing

The authors acknowledge funding from European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska Curie grant agreement No. 956090 (APROPOS: Approximate Computing for Power and Energy Optimisation).

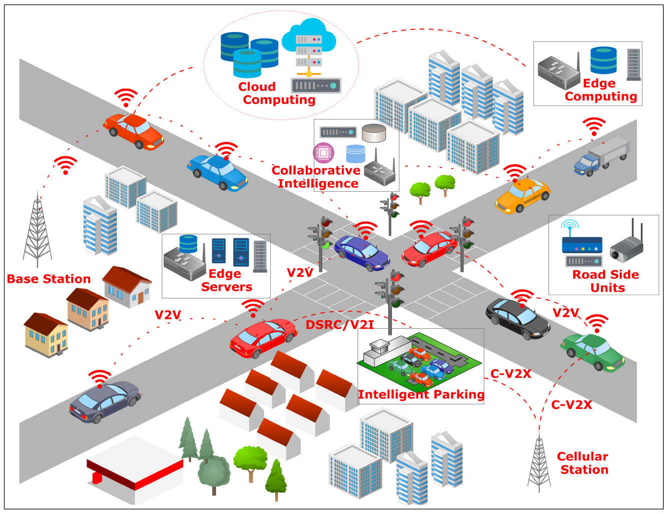


Fig. 1: An overview of vehicle-edge-cloud ecosystem [5]

processing of substantial data from onboard sensors like cameras, radar, lidar, and GPS [3], [6]–[8]. This extensive data handling increases onboard energy consumption, presenting a significant challenge for electric vehicles aimed at sustainable computing [2], [9], [10]. Energy-efficient computing strategies aim to reduce power usage in tasks, while energy-aware methods dynamically optimize energy consumption based on processing and memory workload [11]. Similar techniques in smartphone technology involve advanced power management like dynamic voltage and frequency scaling to enhance battery life without impacting performance [12], [13]. For vehicular systems, transitioning to energy-aware computing necessitates optimizing hardware, software, and computational strategies to adapt dynamically to the vehicle's current energy needs and operational demands [5], [14]. This paper explores the following research questions to develop an adaptive, energy-aware computing framework for connected vehicles.

1) Which computing strategies can enable collaborative and energy-efficient edge deployments in CAV?
2) How can energy-aware techniques be adopted in CAV to optimize energy consumption dynamically?

To address the above questions, we propose a framework to reduce the memory and computational load based on system and application requirements while balancing the accuracy. We also focus on collaborative deployments using edge AI to improve onboard energy efficiency. The framework integrates model partition mechanisms, adaptive model deployment, and model approximation strategies.

II. Related Work and Challenges

Energy-efficient and sustainable computing techniques have gained attention with the increased usage and deployment of autonomous systems, which depend on complex computational models and bring additional energy demands [3], [5], [10]. Connected vehicular technology and applications, which depend on the intersection of computing and communication within the vehicular ecosystem, require energy management and sustainable computing practices. In this section, we discuss existing frameworks that covers energy-efficient strategies and the requirements to advance from energy-efficient to energy-aware computing.

A. Energy-Efficient Computing in CAV

Previous work in energy management includes hardware-level improvements such as advanced power electronics for efficient battery management and power units [2], [15] and software strategies such as data compression and reducing the complexity of models, algorithms [11], [16] to optimize related components. Techniques like model pruning and efficient neural network designs have also helped to reduce the computational demands by using compressed models used in the perception, mapping, and navigation of connected autonomous vehicles [7], [15], [17]. While these methods have effectively reduced model and computational parameters, the compression scope does not consider dynamic vehicular environments' variable computational requirements and characteristics. As a result, energy-efficient strategies operating under static conditions often have less than optimal performance in unpredictable situations where the environmental and operational conditions are dynamic [3]. Existing frameworks such as LoPECS [18] discuss a low-power edge computing system to reduce power consumption in autonomous driving applications. The system uses a heterogeneous computing approach and dynamic task offloading to edge cloudlets, showing onboard energy savings for V2X applications. This framework uses a Heterogeneity-Aware Runtime Layer and a vehicle-edge Coordinator to optimize the user experience with extended battery life and enhanced performance of driving services.

VECMAN [19] introduces energy-aware resource management in vehicular edge computing systems by addressing the challenge of high energy consumption in collaborative applications through optimized resource-sharing and task-offloading strategies. The framework reduces computational energy consumption, using key components such as a resource selector algorithm and an energy manager algorithm, which enhance system reliability and energy efficiency. Using a deep deterministic policy gradient (DDPG) algorithm, a deep reinforcement learning-based framework is discussed for optimizing resource allocation in the Internet of Vehicles [13]. The framework reduces the mobile network operator's energy costs while ensuring timely task completion. This study shows the potential of DRL methods to achieve robust, real-time decision-making in complex vehicular network environments. A carbon-aware framework for AIoT ecosystems focusing on sustainable computing practices is discussed in [10]. The framework discusses energy-efficient communication and low-carbon task offloading. The framework includes a multi-source model for communication and a carbon-aware multi-channel exploration offloading decision algorithm. Experiments show that it outperforms existing methods in reducing data acquisition errors, energy consumption, and carbon emissions, enhancing the overall sustainability of AIoT ecosystems.

B. Energy-Aware Models and Computing

The development of energy-aware computing requires an intelligent strategy that can manage ongoing energy consumption while balancing the performance and quality of CAV applications [3], [16]. Compared to traditional methods focusing on energy efficiency, energy-aware frameworks dynamically adapt to varying energy availability and computational demands. This adaptation involves frameworks that monitor energy usage in real-time for connected vehicles and adjust computational strategies to align with current vehicular needs and application or task priorities, as several CAV applications, such as adaptive cruise control, automatic emergency braking and simultaneous localization and mapping, have strict latency and computing requirements [11], [20].

Strategic computation while balancing system performance is the backbone of energy-aware systems, allowing them to modify operational modes based on real-time environmental data. This can lead to methods and techniques where non-essential computations such as infotainment-related and latency tolerable are scaled down in safe driving conditions to save energy, or essential tasks are prioritized based on safety-critical and latency criteria. Furthermore, the integration of edge computing enhances these systems using distributed computational loads across a network of devices. This arrangement reduces data processing latency and enables localized energy management decisions at the edge level rather than optimizing the complex centralized computing and processing [8], [19], [20]. Using edge computing is also complemented with real-time analytics, reduced latency and adaptive AI technologies to effectively balance performance with energy consumption, resulting in more sustainable vehicular technologies [18].

In energy-aware computing practices, software or model approximation provides a practical and balanced solution by reducing the computational load without compromising performance. Methods such as stochastic rounding, low-precision arithmetic, dynamic precision scaling and probabilistic approximation have been proposed in iot, vehicular and cyber-physical systems [5], [7], [9], [14]. Combined use of these mechanisms can reduce energy consumption while balancing operational performance or reduction in quality and model performance with high energy saving.

III. Components and Framework Overview

The design of the proposed adaptive framework is based on principles discussed in the previous Lopecs framework [18]. The general components of the low-power edge computing system for real-time autonomous driving systems (Lopecs) are discussed in the subsection below, and the following subsection covers our proposed components, which can be integrated with existing Lopecs to advance it from an energy-efficient to an energy-aware framework.

A. Overview of LOPECS Framework

Lopecs is a multi-layered architecture designed to optimize energy consumption and improve computational efficiency in autonomous vehicles. At the first layer, The framework includes input as sensors or incoming data pipelines, followed by the Quality of Experience (QoE) Oriented Service Classification as the first layer, which prioritizes tasks based on their impact on user experience. This layer ensures that safety-critical tasks receive the required computational resources on a priority basis. The next layer is the Runtime Layer, which includes: *Real-Time Operating System (RTOS)* a foundational OS that supports all lower-level operations with minimal latency, *Heterogeneity Aware Scheduler* which manages task allocation across various computing units, ensuring that each task is processed on the appropriate hardware to maximize energy efficiency, and lastly an *OpenCL API + Runtime*, which allows the use of a standardized API for parallel computing across heterogeneous platforms, enhancing the flexibility and efficiency of the system. The third layer described in the Lopecs framework is *Heterogeneous Computing Platform*, which includes computing support for multiple *CPU System* to handle standard computational tasks, with energy-efficient scheduling managed by the Heterogeneity Aware Scheduler, and *GPU Units*, which includes specialized 3D GPUs and an Image Processor, for high-intensity computation tasks such as image processing and complex tasks such as SLAM. This layer also includes support for *Dedicated Accelerators* such as video and audio accelerators, which process specific tasks related to video and audio processing and power usage for multimedia.

Lastly, one of the most essential layers in Lopecs is the Vehicle-Edge Coordinator, which manages the communication between the vehicle's onboard computing system and other edge computing resources (cloudlets). This coordinator dynamically offloads tasks to edge servers based on current network conditions and system load, enhancing the overall energy efficiency by using computing resources available within the vehicle-edge ecosystem. Integrating these components together as a framework addresses autonomous vehicles' onboard power consumption challenge and supports driving services with a distributed, adaptable approach. The evaluation for Lopecs included testing multiple services with a total power consumption of 11W on the Jetson TX1 device. Overall, the framework is the proof-of-concept edge computing system implemented and tested on edge devices with autonomous vehicles, addressing the necessity for real-time, power-efficient computing solutions. While LoPECS included components for reducing power consumption, its performance relies on the proximity of edge cloudlets, which may only be consistently available in some environments. The framework also requires expanding the applicability to a broader range of computing architectures and driving scenarios. Additionally, addressing the problems associated with ML model partitions, data processing mechanisms, and adaptive deployment for inference on vehicle-edge scenarios remains an open challenge.

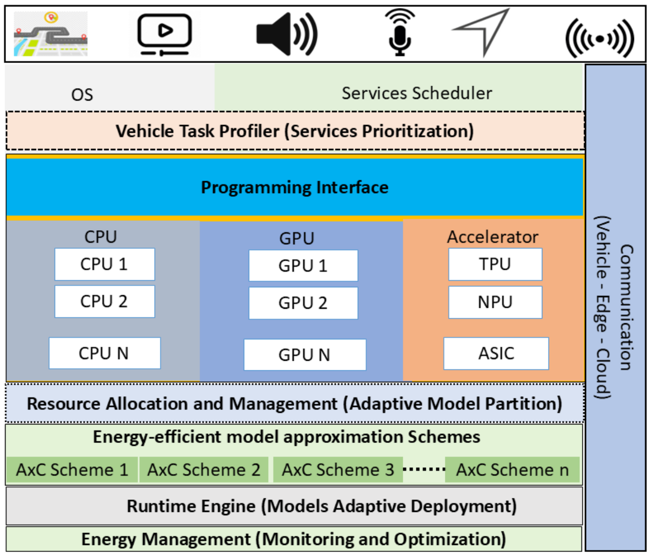

OS
Services Scheduler
Vehicle Task Profiler (Services Prioritization)
Programming Interface
CPU
CPU 1
CPU 2
CPU N
GPU
GPU 1
GPU 2
GPU N
Accelerator
TPU
NPU
ASIC
Resource Allocation and Management (Adaptive Model Partition)
Energy-efficient model approximation Schemes
AxC Scheme 1
AxC Scheme 2
AxC Scheme 3
AxC Scheme n
Runtime Engine (Models Adaptive Deployment)
Energy Management (Monitoring and Optimization)
Communication (Vehicle - Edge - Cloud)

Fig. 2: Proposed adaptive energy-aware framework overview

B. Proposed Components

A high-level overview of the proposed framework is shown in Figure 2. This framework adapts Lopecs five layers and includes these existing layers as a baseline. These five layers used in our adaptive framework have been described as: OS and Service Scheduler, Vehicle task profiler, programming interface, hardware units description, and communication layer. The additional components proposed in this paper are:

1) Model Partition Mechanism: Model partitioning is an important component in our proposed energy-aware framework. This component is designed to distribute large-scale AI models, such as CNN, Vision Transformers (ViTs), across various edge devices efficiently while optimizing energy use and maintaining performance. Our framework introduces the Dynamic Resource-Aware Partitioning (DRAP) strategy, which adapts edge devices' diverse capabilities for tasks. DRAP considers each device's computational power, memory availability, and energy levels; it also analyzes the structure of models such as ViTs to identify optimal points for model splitting. Additionally, it evaluates the latency and accuracy requirements of different CAV tasks to guide partitioning decisions. DRAP aims to enhance the distribution of model layers across devices by solving a multi-objective optimization problem. The goals are to minimize overall energy consumption, balance computational loads, and meet latency requirements for timely operations. The optimization function is:

$$\min \sum_{i=1}^{N} \left(\alpha \frac{E_i}{C_i} + \beta \frac{L_i}{L_{total}} + \gamma T_i \right)$$

where E_i is the energy consumption, C_i the computational capacity, L_i the number of layers assigned to device i, L_{total} the total number of layers, and T_i the processing time on device i. Coefficients α, β, and γ are weighting factors that balance the priorities.

DRAP includes several techniques to reduce complexity and enhance energy efficiency. Dynamic head pruning, for example, adjusts the ViT number of attention heads in ViT layers according to the device's energy state. Elastic dimension scaling reduces the embedding dimensions of layers when energy is limited to decrease computational demands. Selective token processing is employed in resource-constrained scenarios, focusing only on the most critical tokens to minimize computational efforts. Effective communication between devices is essential for energy-aware partitioning. DRAP improves this by applying compression algorithms to feature maps, reducing the volume of data transferred. It also adjusts the precision of transmitted data based on available bandwidth and energy resources. Predictive prefetching uses historical data and current context to predict and prefetch necessary data, reducing latency. DRAP continuously adjusts its strategies based on real-time system conditions. If a device's energy level falls below a certain threshold, tasks are reallocated to devices with more energy. It also dynamically manages workload distribution to avoid overloading any single device. Partitioning strategies are adjusted based on current energy and computing resource conditions, prioritizing critical tasks when needed. These strategies ensure that the framework effectively utilizes resources across heterogeneous edge devices in the vehicular ecosystem, allowing CAVs to use complex AI models efficiently while optimizing energy consumption.

2) Model Approximation Techniques: Our energy-aware framework also integrates advanced model approximation techniques aimed at reducing the energy consumption of onboard systems in connected autonomous vehicles (CAVs) while ensuring the functionality necessary for their operation remains intact. These techniques specifically target the reduction of computational complexity and power demands without substantial impact on the performance of the models [21]. The approximation technique minimises multiplication in convolutional operations, incorporating stochastic methods to minimize energy use. This includes implementing probabilistic kernel application, where convolutional kernels are applied with varying probabilities determined by their relevance to the specific task, thereby avoiding uniform application across all scenarios. We dynamically adjust the sampling rate for these operations to adapt to the system's current energy state. Additionally, sparse activation maps are generated by selectively zeroing out activations probabilistically, which helps to maintain essential features while reducing the computation.

Our framework also uses variational inference techniques optimized for energy saving. Low-rank approximations simplify covariance matrices, reducing memory demands and computational overhead. Adaptive sampling for Monte Carlo estimation allows for adjusting the number of samples based on the system's energy budget, facilitating a balance between accuracy and power consumption. Moreover, hierarchical variational models are implemented to improve inference efficiency by using shared statistical strength across different model layers. Precision in computational tasks is dynamically managed through techniques such as context-aware precision adjustment and mixed-precision quantization. Computational precision is continuously adapted based on external conditions such as vehicle speed and environmental complexity, as well as the importance of the current computational task. This strategy includes layer-specific quantization where critical network layers maintain higher precision, whereas less crucial layers operate with reduced bit-widths. The framework also features an energy-driven reconfiguration system that adjusts computational precision in response to dips below specific energy thresholds, ensuring sustained operation during low-power situations. Additionally, an adaptive floating-point format customizes the number of bits allocated for the exponent and mantissa to match the computational demands precisely.

These approximation strategies are integrated to optimize energy efficiency. The framework actively analyzes incoming tasks to determine the most suitable combination of approximation techniques based on the current energy constraints and task specifics. Real-time performance monitoring maintains a feedback loop that assesses the impact of these approximations on model performance, allowing for dynamic adjustments to ensure an optimal balance between energy efficiency and operational accuracy. Optimization simplifies the model complexity in scenarios where energy is critically low, prioritizing safety-critical functions to maintain essential system integrity. By implementing these model approximation strategies, our framework aims to lower the energy footprint of AI models within CAVs, enabling the deployment of advanced algorithms on resource-constrained edge devices.

3) Adaptive Model Deployment Strategies: Our framework's adaptive model deployment component aims to optimize model orchestration across distributed computing resources, including edge devices and cloud infrastructures. This strategy uses advanced gradient-based techniques to achieve rapid convergence and efficient parallel computation, which is essential for handling dynamic environmental conditions and computational capabilities.

Gradient-Based Optimization Techniques: Our framework uses advanced gradient methods like Enhanced Momentum Gradient Descent (EMGD) to accelerate the convergence of model training processes. EMGD incorporates a momentum term, aiding in navigating along the pertinent directions of the gradient. The equation is given by:

$$\psi^{(t+1)} = \gamma \psi^{(t)} + \nabla \mathrm{Cost}(\Phi^{(t)}), \quad (1)$$

$$\Phi^{(t+1)} = \Phi^{(t)} - \eta \psi^{(t+1)}, \quad (2)$$

where $\psi^{(t)}$ is the velocity, γ is the momentum coefficient, $\nabla\text{Cost}(\Phi^{(t)})$ is the gradient of the cost function, and η is the learning rate. **Asynchronous Gradient Descent (AGD):** This technique allows for asynchronous updates in a parallel computing setup, enhancing scalability and fault tolerance. AGD reduces the need for global synchronization, allowing each node to update based on locally computed gradients:

$$\Phi_{\text{global}}^{(t+1)} = \Phi_{\text{local}}^{(t)} - \eta\nabla\text{Cost}(\Phi_{\text{local}}^{(t)}),$$

where Φ_{global} and Φ_{local} represent the global and local model parameters, respectively. Deployment includes initialization of model parameters and offload decisions, which are iteratively refined using performance and cost metrics. This process ensures optimal configuration based on the computational capabilities of processing units. After this step, the model components are mapped to hardware resources using a lookup table approach. This method ensures that the device handles each model component with the required memory and computing resources, thus optimizing resource use and performance. We use EMGD to map models directly on multiple edge devices, taking advantage of fast convergence properties. AGD synchronizes model parameters across edge devices and the cloud, effectively managing communication delays or failures.

Cloud Deployment: This strategy uses AWS cloud computing instances capabilities to scale computational resources. EMGD improves the training process, and AGD asynchronously updates model parameters across the distributed system, optimizing resource allocation and cost management.

Edge-Cloud Hybrid Deployment: This hybrid approach combines edge computing's low-latency and local data processing with the cloud's computational power. It starts with model training in the cloud using EMGD and distributes model parameters to edge devices via AGD, reducing data transfer requirements and enabling responsive model training. These adaptive deployment strategies ensure the framework meets the computational demands of sophisticated AI models, managing orchestration complexity, optimizing communication overhead, and maintaining rapid convergence across network conditions. This approach is essential for deploying advanced AI models in resource-constrained environments, facilitating more efficient and scalable autonomous vehicle systems.

4) Energy-Aware Systems Using Reinforcement Learning: Advances from energy-efficient to energy-aware systems in connected vehicles require an advanced energy management approach. This transition involves moving from static, predefined energy-saving settings to dynamic, adaptive strategies that can respond in real-time to changing environmental conditions and operational demands. Reinforcement Learning (RL) offers a robust framework for facilitating this shift by enabling continuous learning and adaptation based on the system's interactions with its environment. Reinforcement learning works on the principle of decision-making under uncertainty, where an agent learns to perform actions that maximize some notion of cumulative reward. In the context of connected vehicles, the RL agent interacts with a vehicular environment that is highly dynamic, characterized by continuous changes in driving conditions, network status, and energy availability. The agent's objective is to develop a policy that optimizes energy usage without compromising the main performance metrics such as safety, timeliness, and accuracy of navigational and operational tasks. Reinforcement Learning (RL) provides a robust framework for this shift, enabling continuous learning and adaptation based on the vehicle's interactions with its dynamic environment.

Reinforcement Learning for Dynamic Energy Management: Reinforcement Learning (RL) improves our system from energy-efficient to energy-aware by dynamically optimizing energy use while maintaining essential model and application performance metrics. The RL framework consists of: **State Space:** This includes necessary data about the operational status, including battery levels, processor and memory utilization, latency and network conditions required for making adaptive decisions. **Action Space:** Actions include adjusting computational resource speeds (CPU/GPU), and decisions on task offloading to optimize energy consumption efficiently. **Reward Function:** Designed to balance high performance with low energy consumption, this function penalizes excessive energy use and rewards reductions in energy consumption when a model partition, resource allocation and an approximation strategy is used. The implementation steps are:

Training the RL Agent: Initially, the agent is trained within a simulated environment replicating operational scenarios with diverse memory and computational load. This phase allows the agent to learn optimal policies without real-world effects, using simulated data (model training and evaluation based on batch sizes) to predict energy requirements accurately.

Real-Time Learning and Adaptation: Post-deployment, the RL agent continually refines its policies based on incoming real-world data. This ongoing learning process ensures that the system remains adaptive to changing conditions and can optimize decisions for energy management dynamically.

Integration with Task: The trained RL agent's policies are integrated into the operational applications, for e.g., a perception task, enabling real-time adjustments to the energy management strategies based on the vehicle's current state and its environment.

Model Approximation: These strategies reduce computational demands using probabilistic kernel applications or sparse activation maps, defined as:

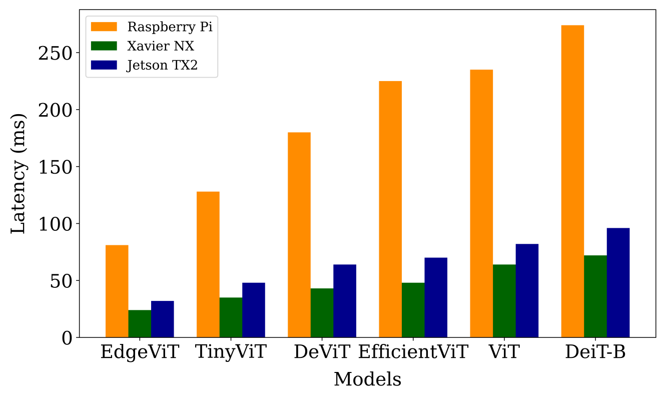
$$E_{approx} = \sum_{i=1}^{n} E_{orig,i} - E_{approx,i}$$

where $E_{orig,i}$ and $E_{approx,i}$ represent the original and reduced energy consumption for each task.

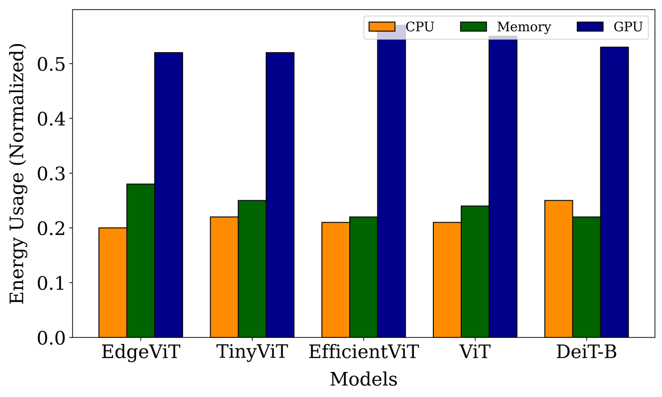
Model Partitioning: This involves distributing computation to optimize energy usage across vehicle, edge-cloud layers:

$$E_{part} = E_{local} + \sum_{j=1}^{m} E_{trans,j} + E_{remote,j}$$

Here E_{local}, $E_{trans,j}$, and $E_{remote,j}$ are used for local computations and data transmission energies.



(a) Latency (ms)



(b) Energy Usage

Fig. 3: Latency and Energy Usage Comparison for Distributed Models

Model Deployment Mechanisms: Both synchronous and asynchronous mechanisms are used to process data, influencing overall energy consumption:

$$E_{deploy} = \alpha E_{sync} + (1 - \alpha) E_{async}$$

Where α refers to the operations using synchronous methods. Utilizing RL in this capacity enhances overall energy management in connected vehicles, dynamically adjusting operational strategies to maintain optimal performance levels while minimizing energy consumption. This approach aligns with sustainability goals and improves the deployment strategy of advanced AI applications in energy-limited scenarios for autonomous and efficient vehicular technologies. By implementing these strategies, our framework improves the energy efficiency of connected autonomous vehicles to operate efficiently and sustainably under varying operational conditions.

C. Integration of Proposed Components

Integrating the proposed components: resource allocation & management, approximation scheme, adaptive model deployment, and energy management into the LOPECS framework improves its capabilities from energy-efficient to energy-aware computing. The resource allocation-management module interfaces with the previous heterogeneous-aware scheduler, optimizing the distribution of computational tasks across available resources while considering energy constraints. The approximation scheme component works in coordination with the computing unit platform, dynamically adjusting the precision of operations based on the capabilities of each computing unit and current energy levels. Adaptive model deployment uses the vehicle-edge coordinator to intelligently partition and distribute AI models across vehicle and edge resources, minimizing data transfer and energy consumption. Finally, the energy management module integrates with the task profiler, continuously monitoring energy usage and providing output to components for real-time optimization. This integration enables the framework to make energy-aware decisions at different stages of task execution, from initial profiling to final deployment, resulting in a more efficient and adaptable system for vehicle-edge services.

TABLE I: Comparison of baseline and our proposed method.

Models	Methods	mAP (%)	Latency	Energy (mJ)
EdgeViT	Central	57.3	4.25	370.6
	Ours	55.6	3.88	290.7
TinyViT	Central	55.7	3.70	430.6
	Ours	51.1	3.27	367.5
EfficientViT	Central	53.9	3.86	462.7
	Ours	51.7	3.34	381.6
ViT	Central	49.7	4.73	493.6
	Ours	42.5	3.02	402.5
DeiT-B	Central	46.6	5.05	432.0
	Ours	40.2	3.54	341.4

IV. Test and Evaluation

Hardware: For evaluations the devices used includes Xavier NX, Jetson TX2, and Raspberry Pi 4, which differ in computational power and memory capabilities. This allowed us to evaluate the flexibility of our partitioning strategies in the real edge AI conditions. We measured each device's energy use and latency during the inference phases.

Vision Transformer Models: We used several Vision Transformer models with architecture, including attention mechanisms, feed-forward network (FFN) layers, and the integration of various operations from CNNs, GNNs, and MLPs. Our evaluation included: *ViT:* The standard Vision Transformer model applying transformers to image recognition with self-attention mechanisms. *EdgeViT:* Adapted for edge computing, this model has computational efficiency and minimizes memory demands for real-time image processing on constrained devices. *TinyViT:* This model offers a compact solution for environments with limited resources, balancing the size and computational requirement against accuracy. *DeiT-B:* Optimizes training efficiency through knowledge distillation, ideal for limited data environments. *EfficientViT:* A ViT model designed for efficient inference on edge devices, optimizing the balance between model performance and devices.

Dataset: The models were tested on the OPV2V dataset [22], which supports cooperative perception tasks. Generated using the OpenCDA framework and CARLA simulator, this dataset has around 73 diverse driving scenes in 9 cities with 12K frames of LiDAR point clouds and RGB camera images, providing a robust platform for evaluating our models' performance in autonomous driving applications.

Evaluation Metrics: We use onboard *Energy Consumption*, monitored using inbuilt and external tools to measure power usage during inference. *mAP*, is used for benchmarking 3D object detection tasks. *Latency*, is used to measure the time required to complete inference across the distributed devices, as shown in Table I, which compares our method against a centralized approach. The table shows that models like EdgeViT and TinyViT provide high efficiency with lower energy demands. Figure 3 shows the model's latency and normalized energy usage across devices, which helps to identify the computational challenges and processing requirements on the Raspberry Pi and powerful Jetson devices. For normalized energy tracking, we used NVIDIA's NVML for GPUs and Tegrastats for Tegra processors, enabling us to monitor and optimize energy consumption. CPU consumption ranges from 0.22 for EfficientViT to 0.29 for ViT, while memory from 0.26 for EdgeViT to 0.34 for both ViT and DeiT-B. The GPU energy usage varies from 0.37 for ViT to 0.55 for TinyViT. This analysis shows that models such as EdgeViT, TinyViT, and EfficientViT have efficient GPU processing. While the legacy ViT model shows a more balanced distribution of energy consumption across components, showing potential for further optimization of components.

V. Conclusion and Future Work

This study proposes an adaptive framework that includes model partition mechanisms, adaptive model deployment, and model approximation strategies to balance energy efficiency with model performance and accuracy using an energy-aware approach. We have described the proposed components, their integration and testing using vision models to answer the mentioned research questions, which shows the potential for onboard energy savings and balanced performance in the vehicular ecosystem. Future work will test the framework across heterogeneous edge devices, including accelerators and multi-modalities, to validate its adaptability and effectiveness in diverse operational environments. Additionally, the scope of vehicular applications will be expanded to include more complex computational tasks such as HD mapping or video streaming, where some computational processes are offloaded and shared with cloud environments, which can help explore the balance between edge and cloud computing, aiming to optimize resource allocation and further reduce the energy footprint of connected vehicular systems.

References

[1] D. Cao, X. Wang, L. Li, C. Lv, X. Na, Y. Xing, X. Li, Y. Li, Y. Chen, and F.-Y. Wang, "Future directions of intelligent vehicles: Potentials, possibilities, and perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 1, pp. 7–10, 2022.

[2] T. Aldhanhani, A. Abraham, W. Hamidouche, and M. Shaaban, "Future trends in smart green iov: Vehicle-to-everything in the era of electric vehicles," *IEEE Open Journal of Vehicular Technology*, 2024.

[3] A. N. Houssam Kanso and E. Exposito, "A review of energy aware cyber-physical systems," *Cyber-Physical Systems*, vol. 10, no. 1, pp. 1–42, 2024.

[4] G. Xu, Z. Hao, Y. Luo, H. Hu, J. An, and S. Mao, "Devit: Decomposing vision transformers for collaborative inference in edge devices," *IEEE Transactions on Mobile Computing*, 2023.

[5] D. Katare and A. Y. Ding, "Energy-efficient edge approximation for connected vehicular services," in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2023, pp. 1–6.

[6] D. Katare and M. El-Sharkawy, "Real-time 3-d segmentation on an autonomous embedded system: using point cloud and camera," in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*. IEEE, 2019, pp. 356–361.

[7] A. Ravi, V. Chaturvedi, and M. Shafique, "Vit4mal: Lightweight vision transformer for malware detection on edge devices," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5s, pp. 1–26, 2023.

[8] S. Kumari, A. R. Hota, and S. Mukhopadhyay, "Data-driven robust optimization for energy-aware safe motion planning of electric vehicles," *IEEE Transactions on Intelligent Vehicles*, 2024.

[9] W. Gao and X. Yang, "A joint resource allocation and task offloading scheme for energy-aware and latency constrained vehicular edge computing network," in *2024 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2024, pp. 167–172.

[10] Z. Song, M. Xie, J. Luo, T. Gong, and W. Chen, "A carbon-aware framework for energy-efficient data acquisition and task offloading in sustainable aiot ecosystems," *IEEE Internet of Things Journal*, 2024.

[11] C. Liang, Y. Zhao, Z. Gao, K. Cheng, B. Wang, and L. Huang, "Ealso: joint energy-aware and latency-sensitive task offloading for artificial intelligence of things in vehicular fog computing," *Wireless Networks*, pp. 1–17, 2024.

[12] P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, and K. Li, "A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–44, 2020.

[13] X. Kong, G. Duan, M. Hou, G. Shen, H. Wang, X. Yan, and M. Collotta, "Deep reinforcement learning-based energy-efficient edge computing for internet of vehicles," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6308–6316, 2022.

[14] H. J. Damsgaard, A. Grenier, D. Katare, Z. Taufique, S. Shakibhamedan, T. Troccoli, G. Chatzitsompanis, A. Kanduri, A. Ometov, A. Y. Ding *et al.*, "Adaptive approximate computing in edge ai and iot applications: A review," *Journal of Systems Architecture*, p. 103114, 2024.

[15] Y.-C. Wang, J. Xue, C. Wei, and C.-C. J. Kuo, "An overview on generative ai at scale with edge-cloud computing," *IEEE Open Journal of the Communications Society*, 2023.

[16] C. Lin, C. Li, and J. Liu, "A qos-aware training framework for vit compression, partition, and distillation," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–2.

[17] H. Liang, Q. Sang, C. Hu, D. Cheng, X. Zhou, D. Wang, W. Bao, and Y. Wang, "Dnn surgery: Accelerating dnn inference on the edge through layer partitioning," *IEEE transactions on Cloud Computing*, vol. 11, no. 3, pp. 3111–3125, 2023.

[18] J. Tang, S. Liu, L. Liu, B. Yu, and W. Shi, "Lopecs: A low-power edge computing system for real-time autonomous driving services," *IEEE Access*, vol. 8, pp. 30 467–30 479, 2020.

[19] T. Bahreini, M. Brocanelli, and D. Grosu, "Vecman: A framework for energy-aware resource management in vehicular edge computing systems," *IEEE Transactions on Mobile Computing*, 2021.

[20] A. B. Sada, A. Khelloufi, A. Naouri, H. Ning, and S. Dhelim, "Energy-aware selective inference task offloading for real-time edge computing applications," *IEEE Access*, 2024.

[21] D. Katare, E. Marin, N. Kourtellis, M. Janssen, and A. Y. Ding, "Arasec: Adaptive resource allocation and model training for serverless edge computing," *IEEE Internet Computing*, 2024.

[22] R. Xu, H. Xiang, X. Xia, X. Han, J. Li, and J. Ma, "Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2583–2589.